НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

«**До захисту допущено**»
В.о. завідувача кафедри
_____ __*О.А.Павлов*__
(підпис) (ініціали, прізвище)
“___” _____ 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки *6.050103 «Програмна інженерія»*

спеціальність *«Програмне забезпечення систем»*

на тему: *GSM сигналізація з веб-сервером*

Виконав: студент 4 курсу, групи ІП-52

Шуліков Дмитро Дмитрович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *доц. к.т.н. Ліщук К.І.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант з графічної документації *доц. к.т.н. Ліщук К.І.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент *доц. каф.ТК. к.т.н.,доц Лісовиченко О.І.* _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – *6.050103 «Програмна інженерія» (Програмне забезпечення систем)*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Шулікову Дмитру Дмитровичу
(прізвище, ім`я, по батькові)

1. Тема проекту «*GSM сигналізація з веб-сервером»*

керівник проекту *Ліщук Катерина Ігорівна, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “23” квітня 2019 р. №1181-с

2. Термін подання студентом проекту «03» *червня 2019 року*

3. Вихідні дані до проекту
Технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: опис предметної області, аналіз, аналіз існуючих проектів, розроблення функціональних та нефункціональних вимог
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, архітектура програмного забезпечення. Конструювання програмного забезпечення, аналіз безпеки даних
3) Аналіз якості та тестування програмного забезпечення
4) Впровадження та супровід програмного забезпечення

5. Перелік графічного матеріалу

1) Схема електрична принципова

2) Схема електрична структурна

3) Креслення вигляду екранних форм

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «12» *березня 2018 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	*Вивчення предметної області*	18.03.2019	
2	*Аналіз існуючих методів розв'язання задачі*	23.03.2019	
3	*Постановка та формалізація задачі*	25.03.2019	
4	*Аналіз вимог до програмного забезпечення*	01.04.2019	
5	*Моделювання програмного забезпечення*	08.04.2019	
6	*Розробка архітектури програмного забезпечення*	15.04.2019	
7	*Розробка програмного забезпечення*	30.04.2019	
8	*Оформлення пояснювальної записки*	17.05.2019	
9	*Виконання графічних документів*	23.05.2019	
10	*Подання ДП на попередній захист*	28.05.2019	
11	*Подання ДП рецензенту*	03.05.2019	
12	*Подання ДП на основний захист*	10.06.2019	

Студент _____ Шуліков Д.Д.
(підпис)

Керівник проекту _____ Ліщук К.І.
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проект	2	
2	А4	КПІ.ІП-5224.045440.01.81	Пояснювальна записка	75	
3	А4	КПІ.ІП-5224.045440.02.91	Технічне завдання	11	
6	А3	КПІ.ІП-5224.045440.03.99 СП	Схема електрична принципова	1	
7	А3	КПІ.ІП-5224.045440.03.99 СС	Схема електрична структурна	1	
8	А3	КПІ.ІП-5224.045440.03.99 КЕ	Креслення вигляду екранних форм	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-5224.045440.00.90		
Розробн.		Шуліков Д.Д.			Відомість дипломного проекту	Літ.	Лист 1 / Листів 1
Керівн.		Ліщук К.І.					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-52	
Н/контр.		Ліщук К.І.					
В.о.зав.каф.		Павлов О.А.					

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з чотирьох розділів, містить 21 таблицю, 10 рисунків, 1 додаток та 10 джерел – загалом 75 сторінок.

Об`єкт дослідження: веб застосунок з реляційною базою даних.

Мета дипломного проекту: створити систему віддаленого керування пристроєм домашньої сигналізації та власне сам пристрій.

У першому розділі було проаналізовано предметну модель розробки та розглянуто загальні вимоги до системи.

У другому розділі було розроблено загальну архітектуру системи. Побудовані діаграми BPMN, розглянутий список класів та методів системи.

У третьому розділі проаналізовано значимість тестування для розробки, розроблено план тестування проекту.

У четвертому розділі було описано процес розгортання веб застосунку.

КЛЮЧОВІ СЛОВА: СИГНАЛІЗАЦІЯ, GSM, ВІДДАЛЕНИЙ СЕРВІС

ABSTRACT

Explanatory note of the diploma project consists of 4 sections, 10 illustrations, 1 annex, 21 tables and 10 sources – total 75 pages.

The object of study: server application with relational database.

The aim of the diploma project: create a remote control system for the home alarm device and the device itself.

In the first section the subject area of development was analyzed and the general system requirements were considered.

In the second section, the general architecture of the system was developed. BPMN charts were built. Classes and methods of the system were described.

The third section analyzes the significance of testing for development and develops a project testing plan.

The fourth section describes how to deploy the web application.

KEYWORDS: ALARM, GSM, REMOTE SERVICE

Пояснювальна записка
до дипломного проекту

на тему: GSM сигналізація з веб-сервером

Київ – 2019 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Spring Framework – це готовий фреймворк з відкритим вихідним кодом для розробки веб-застосунків на мові Java.

Мікроконтролер – комп’ютер з одною мікросхемою, здатний виконувати прості завдання.

Ethernet Shield – плата, яка дає змогу розширити мікроконтролер та підключити його до мережі.

GSM – це стандарт для мобільного зв’язку, який забезпечує шифрування з відкритим ключем.

GSM модуль – плата, яка дає змогу розширити мікроконтролер та надати йому можливість використовувати стільниковий зв’язок.

ВСТУП

Важко навіть уявити, на скільки сильно змінилось життя людства за останні 20-30 років. Ми живемо за часу сплеску розвитку інформаційних технологій, можливості яких тепер доступні для всього населення планети. Використання цих можливостей вже давно вийшло за рамки науки та армії, особливо після появи загальної світової павутини – Інтернету. Тепер використання цих ресурсів життєво необхідно для лікаря, вчителя, охоронця, школяра та практично для будь-якої людини незалежно від її роду діяльності.

Паралельно стрімко зростають й інші галузі, які впроваджують у своїх рішеннях інструменти, що виникли у наслідок згаданого вище зросту рівня інформаційних технологій, наприклад, системи безпеки.

Сучасні охоронні системи все більш надійні та все менш складні для використання навіть школярем. Вони йдуть у ногу з наукою, щоб не дати випереджати себе зловмисникам, хоча, на жаль, таке іноді й трапляється.

Мета розробки – зменшення часу реакції на можливі загрози для безпеки будинку за рахунок використання мережі Інтернет, а також зменшення собівартості кінцевої системи.

Завданням даної роботи є розробка системи домашньої сигналізації. Компонентами розроблюваної системи є серверний застосунок, база даних та власне пристрій сигналізації. Результатом роботи є система для надання можливості власнику пристрою встановити вдома сигналізацію та керувати нею віддалено за допомогою ВЕБ-застосунку.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Розробляючи систему охорони для дому потрібно чітко розуміти відповідальність і необхідну уважність до деталей реалізації проекту, так як така система має критичний рівень вимог до інформаційної безпеки. Водночас система повинна мати інтуїтивно зрозумілий інтерфейс та простоту встановлення пристрою сигналізації.

1.2 Змістовний опис і аналіз предметної області

Перш за все система повинна мати змогу аутентифікувати користувача та працювати відповідно до наданої ним конфігурації. Головна функція – можливість для користувача змінювати стан сигналізації дистанційно, тобто вмикати та вимикати моніторинг, тож потрібно надати користувачу таку можливість після авторизації. Для підвищення надійності потрібно забезпечити декілька каналів сповіщення, такі як, наприклад, сповіщення у веб-додатку та SMS сповіщення з GSM модулю. Необхідно мати змогу отримати інформацію про стан системи на даний момент, наприклад, стан моніторингу та заряд акумулятору. Також може бути корисною можливість надавати користувачу статистику системи, яка буде зберігатись у базі даних.

1.3 Аналіз успішних ІТ-проектів

Сьогодні на ринку охоронних сигналізацій представлено безліч різних варіантів, які відрізняються не тільки функціональністю й зовнішнім виглядом, але й ціною.

З розвитком послуг стільникового зв'язку все більше розповсюдження отримують GSM сигналізації. GSM сигнал на відміну від звичайного радіосигналу не потребує станції с великим рівнем споживання електроенергії для його розповсюдження. Для його роботи необхідне мінімальний рівень

живлення та підключення до будь-якого оператора стільникового зв'язку. Тож такі системи цілком справедливо швидко заволоділи ринком.

На ринку України існує багато постачальників охоронних систем з GSM модулями. Можна перераховувати такі як: Хорт, Express GSM, WOFEA WiFi, GSM-Kit-30, Ajax StarterKit. Усі вони мають більш-менш схожий функціонал, але доволі високу, на відміну від мого проекту, ціну та не схильні до розширення іншими пристроями розумного дому.

GSM сигналізація Хорт має достатній температурний діапазон для роботи -25С – +45С, використовує GSM модуль «SIM900R» компанії «SIM Technology» та має на борту контролер компанії «STMicroelectronics». Час автономної роботи при стандартній комплектації – до 6 діб.

Сигналізація Express GSM не потребує зовнішнього джерела живлення та працює від 3 батарейок типу «АА» до 12 місяців. Керування сигналізацією здійснюється за допомогою пульту. Також доступна сенсорна клавіатура. Сигналізація має сирену. Виявлення порушення безпеки працює завдяки інфрачервоному датчику. Для функціонування системи необхідна постійно доступна зона стільникового покриття. Температурний діапазон для коректної роботи -30С – +50С. Вологість не більша 93%. Час сповіщення приблизно 15-30 секунд. Сигналізація має можливість запису короткого голосового повідомлення.

Бездротова сигналізація WOFEA WiFi V10 має функцію керування через додаток для гаджетів. Також доступне керування через SMS повідомлення, дзвінки, пульт дистанційного керування. Має змогу підтримувати від 1 до 99 датчиків. Працює через з'єднання с Wi-Fi мережею. У базовій комплектації має датчик руху, датчик відчинення вікон та дверей, сирену.

Комплект GSM-Kit-30 тримає зв'язок з системою через GSM канал. У разі спрацювання сигналізації – спрацьовує сирена та викликаються номера запрограмованих 6 абонентів. Має встановлений акумулятор для підтримки роботи при втраті зовнішнього живлення. Керування системою здійснюється

клавіатурою на центральному блоці або дистанційним пультом. Має вбудований годинник та LCD дисплей.

Ajax StarterKit – сигналізація більш високого цінового сегменту. До стартового набору для захисту приміщень входить хаб (головний контролер домашньої системи), датчик руху, датчик відкритих дверей та пульт керування. Пристрої працюють до 7 років без заміни акумуляторів. Система працює по протоколу Jeweller та має резервний GSM канал зв’язку.

1.4 Аналіз вимог до програмного забезпечення

Варіанти використання в системі представлені нижче:

Таблиця 1.1 – Варіант використання UC001

Назва	Авторизація користувача
Опис	Користувач має змогу пройти авторизацію
Учасники	Користувач
Передумови	Користувач має створений обліковий запис
Постумови	Користувач проходить авторизацію
Основний сценарій	Користувач звертається до сторінки логіну, яка має поля «логін» та «пароль». Користувач заповнює поля та натискає кнопку «Увійти». Користувач проходить авторизацію
Розширення сценаріїв	4.1 Система виявляє, що внесені дані не є коректними. 4.2 Система надає користувачу повідомлення про помилку

Таблиця 1.2 – Варіант використання UC002

Назва	Створення користувача
Опис	Користувач з роллю адміністратора має права на створення нових облікових записів користувачів
Учасники	Користувач
Передумови	Користувач має створений обліковий запис з роллю адміністратора
Постумови	Система має новий обліковий запис для нового користувача
Основний сценарій	Користувач звертається до сторінки створення нового користувача, яка містить поля «Логін», «Пароль» та кнопку «Зареєструвати». Користувач заповнює поля та натискає на кнопку
Розширення сценаріїв	4.1 Система виявляє, що користувач з таким ім’ям вже створений. 4.2 Система надає користувачу повідомлення про існуюче ім’я

Таблиця 1.3 – Варіант використання UC003

Назва	Вихід з облікового запису
Опис	Користувач має змогу вийти з облікового запису
Учасники	Користувач
Передумови	Користувач авторизований у системі

Продовження таблиці 1.3

Постумови	Користувач вийшов з облікового запису
Основний сценарій	Користувач на головній сторінці натискає кнопку «Вийти з облікового запису»
Розширення сценаріїв	

Таблиця 1.4 – Варіант використання UC004

Назва	Видалення користувача
Опис	Користувач з роллю адміністратора має права на видалення існуючих облікових записів користувачів
Учасники	Користувач
Передумови	Користувач має створений обліковий запис з роллю адміністратора
Постумови	Видалений обліковий запис користувача
Основний сценарій	Користувач з роллю адміністратора звертається до сторінки зі списком існуючих користувачів. Біля користувача, якого потрібно видалити, натискає на кнопку «Видалити»
Розширення сценаріїв	

Таблиця 1.5 – Варіант використання UC005

Назва	Увімкнення сигналізації
Опис	Користувач має можливість увімкнути сигналізацію
Учасники	Користувач
Передумови	Користувач має створений обліковий запис та під’єднаний пристрій сигналізації
Постумови	Сигналізація увімкнена
Основний сценарій	На сторінці керування сигналізацією користувач натискає кнопку «Увімкнути сигналізацію»
Розширення сценаріїв	4.1 Система виявляє, що пристрій не було під’єднано до системи 4.2 Система демонструє повідомлення про непід’єднаний пристрій

Таблиця 1.6 – Варіант використання UC006

Назва	Вимкнення сигналізації
Опис	Користувач має можливість вимкнути сигналізацію
Учасники	Користувач
Передумови	Користувач має створений обліковий запис та під’єднаний пристрій сигналізації
Постумови	Сигналізація вимкнена

Продовження таблиці 1.6

Основний сценарій	На сторінці керування сигналізацією користувач натискає кнопку «Вимкнути сигналізацію»
Розширення сценаріїв	4.1 Система виявляє, що пристрій не було під’єднано до системи 4.2 Система демонструє повідомлення про непід’єднаний пристрій

Таблиця 1.7 – Варіант використання UC007

Назва	Запит стану батареї
Опис	Користувач може зробити запит стану батареї живлення сигналізації
Учасники	Користувач
Передумови	Користувач має обліковий запис у системі та під’єднаний пристрій сигналізації
Постумови	Користувач має змогу бачити актуальний стан батареї живлення сигналізації
Основний сценарій	На сторінці керування сигналізацією користувач натискає кнопку «Отримати стан батареї»
Розширення сценаріїв	4.1 Система виявляє, що пристрій не було під’єднано до системи 4.2 Система демонструє повідомлення про непід’єднаний пристрій

Таблиця 1.8 – Варіант використання UC008

Назва	Перегляд статистики користувача
Опис	Користувач має змогу переглянути свою статистику користування сигналізацією
Учасники	Користувач
Передумови	Користувач має створений обліковий запис
Постумови	Користувач бачить свою статистику користування сигналізацією
Основний сценарій	Користувач звертається до сторінки перегляду статистики
Розширення сценаріїв	

Таблиця 1.9 – Варіант використання UC009

Назва	Отримання сигналу про спрацювання сигналізації
Опис	Користувач повинен отримати сигнал при спрацюванні сигналізації
Учасники	Користувач
Передумови	Користувач має обліковий запис в системі та увімкнену сигналізацію
Постумови	Користувач отримує SMS з повідомленням про спрацювання сигналізації

Продовження таблиці 1.9

Основний сценарій	Пристрій сигналізації при спрацюванні за допомогою GSM модулю відправляє SMS повідомлення власнику
Розширення сценаріїв	

Також система має такі функціональні вимоги:

Таблиця 1.10 – Опис функціональної вимоги REQ001

Номер	REQ001
Назва	Створення нових користувачів у системі
Опис	Система надає можливість користувачу з роллю адміністратора створити новий обліковий запис у системі

Таблиця 1.11 – Опис функціональної вимоги REQ002

Номер	REQ002
Назва	Видалення існуючих користувачів у системі
Опис	Система надає можливість користувачу з роллю адміністратора видалити існуючий обліковий запис у системі

Таблиця 1.12 – Опис функціональної вимоги REQ003

Номер	REQ003
Назва	Вхід в обліковий запис
Опис	Система надає можливість користувачу пройти авторизацію

Таблиця 1.13 – Опис функціональної вимоги REQ004

Номер	REQ004
Назва	Вихід з облікового запису
Опис	Система надає можливість користувачу, який пройшов авторизацію, вийти з облікового запису

Таблиця 1.14 – Опис функціональної вимоги REQ005

Номер	REQ005
Назва	Увімкнення сигналізації
Опис	Система має функціонал для увімкнення сигналізації користувачем, який має обліковий запис та під’єднаний пристрій сигналізації

Таблиця 1.15 – Опис функціональної вимоги REQ006

Номер	REQ006
Назва	Вимкнення сигналізації
Опис	Система має функціонал для вимкнення сигналізації користувачем, який має обліковий запис та під’єднаний пристрій сигналізації

Таблиця 1.16 – Опис функціональної вимоги REQ007

Номер	REQ007
Назва	Запит стану батареї

Продовження таблиці 1.16

Опис	Система має функціонал для отримання стану батареї з пристрою користувачем, що має створений обліковий запис

Таблиця 1.17 – Опис функціональної вимоги REQ008

Номер	REQ008
Назва	Перегляд статистики користувача
Опис	Система має функціонал для перегляду статистики користувача, який має обліковий запис у системі

Таблиця 1.18 – Опис функціональної вимоги REQ009

Номер	REQ009
Назва	Отримання сповіщення про спрацювання сигналізації
Опис	Система має функціонал для сповіщення користувача про спрацювання сигналізації

Таблиця 1.19 – Опис функціональної вимоги REQ010

Номер	REQ010
Назва	Під’єднання пристрою сигналізації до системи
Опис	Система має функціонал для створення зв’язку між додатком та пристроєм сигналізації

Побудуємо результуючу матрицю трасування на Рисунку

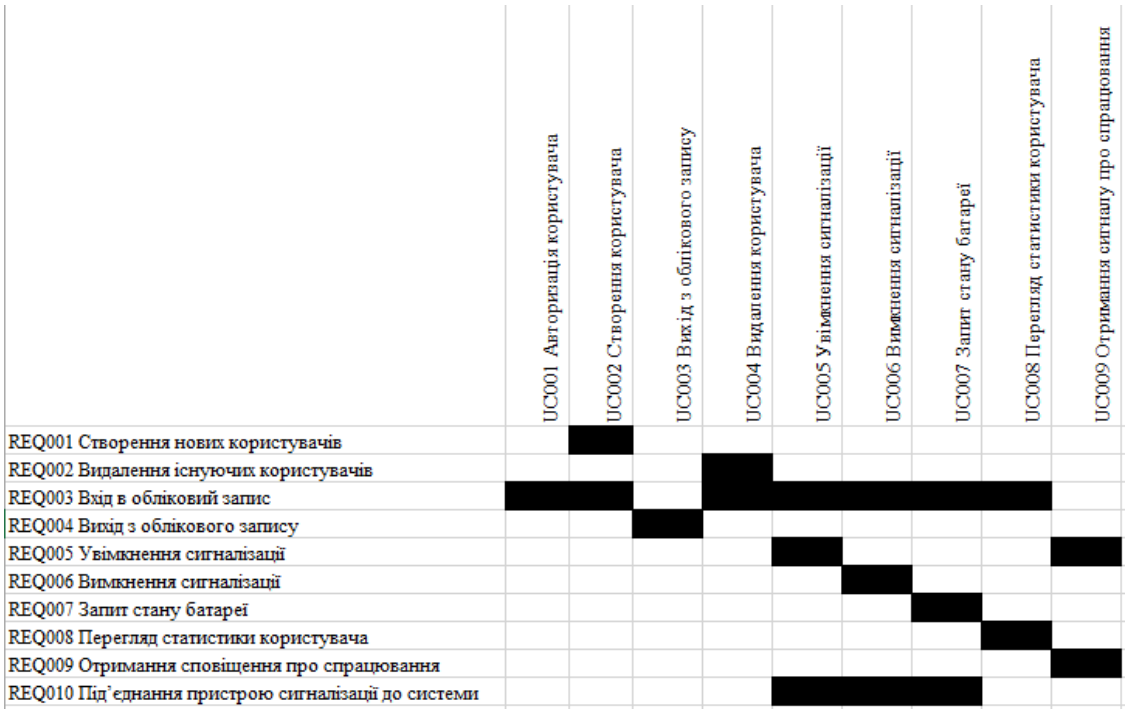


Рисунок 1.1 – Матриця трасування

1.5 Постановка задачі

Основним призначенням створюваної системи є зменшення часу реакції користувачів на порушення безпеки їх приватної власності та зменшення собівартості інструменту для цього.

Мета розробки – застосувати комплексні знання наукових дисциплін, здобуті у процесі навчання, для побудови складної охоронної системи.

Задачі:

- авторизація користувачів у системі;
- дистанційне керування пристроєм сигналізації;
- моніторинг стану пристрою;
- збір статистики користувача;
- безперебійна робота пристрою;

- збереження надійності шифрування для даних;
- зменшення собівартості кінцевої системи.

1.6 Висновки по розділу

У розділі було ретельно проаналізовано предметну область розробки та функціональні вимоги проекту. Також були розглянуті успішні проекти в області охоронних систем та можливості їх удосконалення. Було представлено основне призначення розробки, її мета та задачі.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Для створення програмного забезпечення необхідно провести детальний аналіз та моделювання, для чого буде використано методологію створення діаграм BPMN та буде відображено основні процеси використання проекту користувачем.

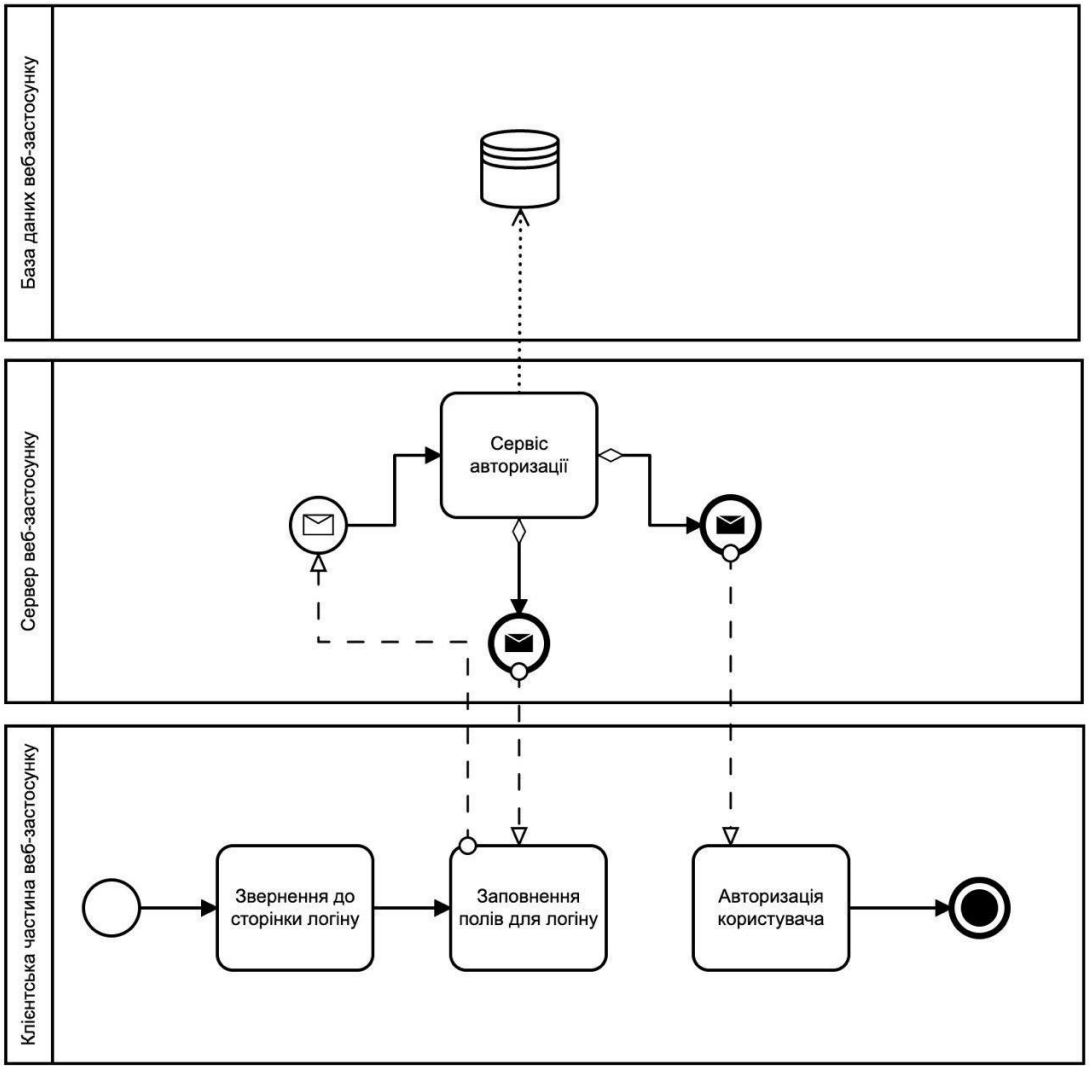


Рисунок 2.1 – Авторизація користувача у системі

Послідовний опис процесу авторизації користувача у системі:

- користувач звертається до сторінки логіну;
- вводить дані для логіну;

– відбувається обробка даних на сервері;

– сервер звертається до бази даних;

– якщо вони коректні – то авторизація проходить успішно;

– якщо некоректні, то користувач повертається на сторінку логіну.

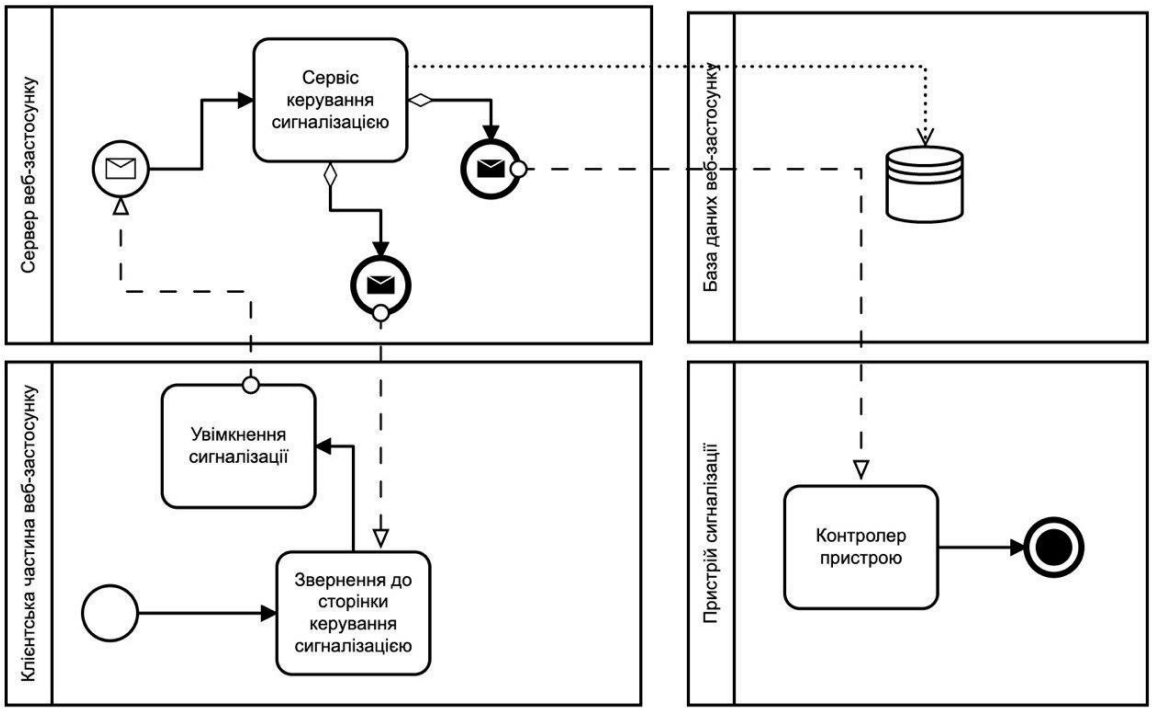


Рисунок 2.2 – Увімкнення сигналізації

Опис процесу увімкнення сигналізації:

– користувач звертається до сторінки керування сигналізацією;

– натискає на кнопку увімкнення сигналізації;

– відправляє на сервер запит на вмикання;

– сервіс керування сигналізацією перевіряє з’єднання з пристроєм та надсилає запит на вмикання сигналізації контролеру пристрою;

– якщо пристрій під’єднаний, то сигналізація успішно вмикається;

– інакше сервіс повідомляє користувача, що пристрій не під’єднано, повертаючи на сторінку керування сигналізацією.

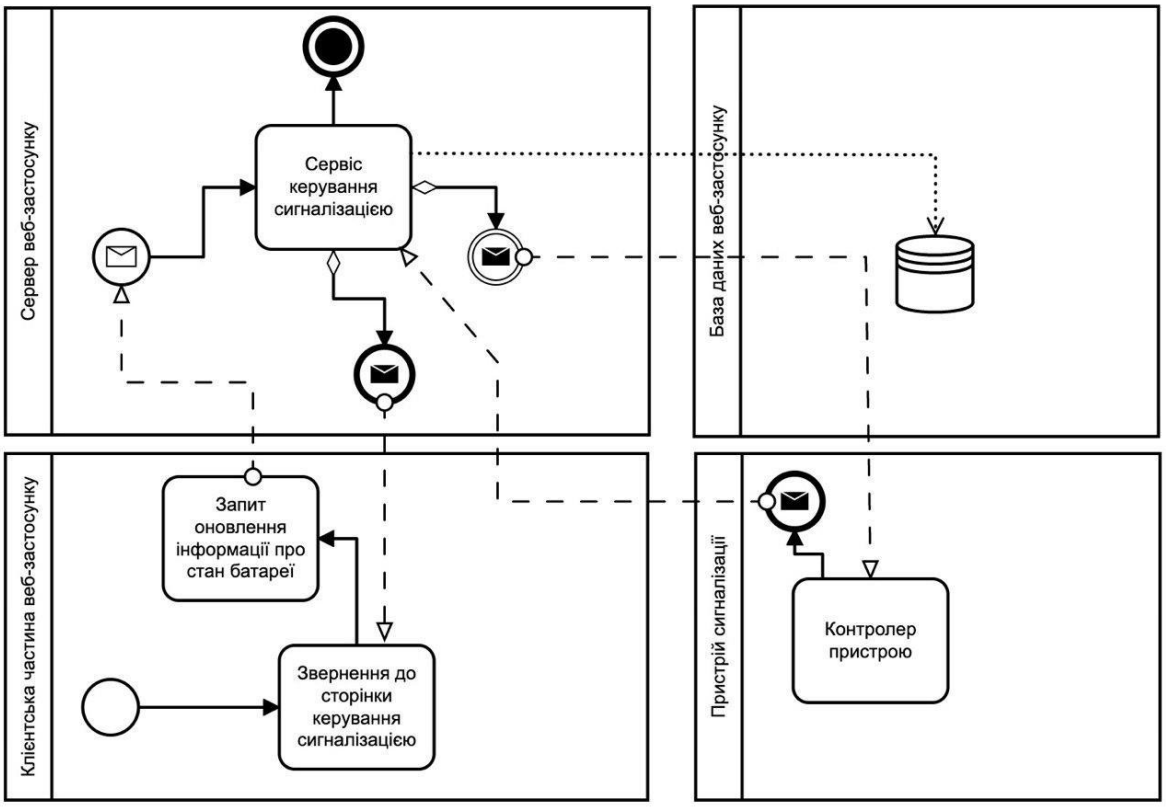


Рисунок 2.3 – Оновлення інформації про батарею

Послідовний опис процесу оновлення інформації про стан батареї:

– звернення до сторінки керування сигналізацією;

– запит на оновлення інформації про стан батареї;

– сервіс керування сигналізацією перевіряє з’єднання з пристроєм та надсилає запит на отримання інформації про стан батареї;

– якщо пристрій під’єднаний, то його контролер відправляє стан батареї до сервісу веб-застосунку, який зберігає стан батареї у базі даних;

– інакше сервіс повідомляє користувача, що пристрій не під’єднано, повертаючи на сторінку керування сигналізацією.

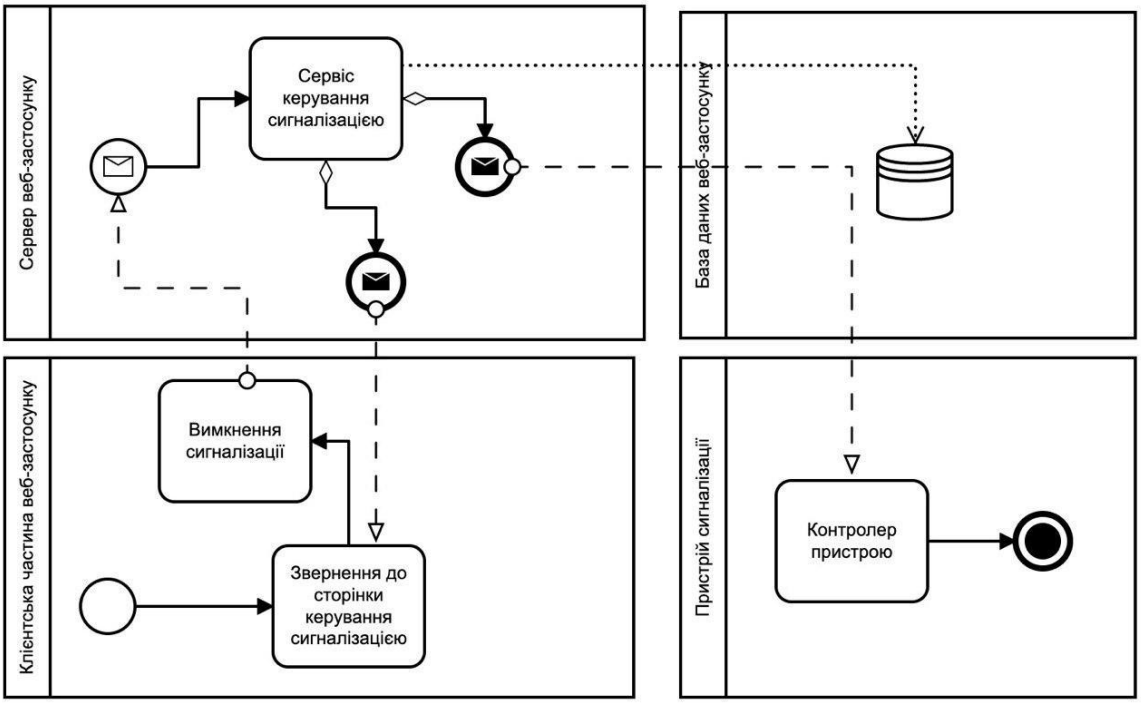


Рисунок 2.4 – Вимкнення сигналізації

Опис процесу вимкнення сигналізації:

– користувач звертається до сторінки керування сигналізацією;

– натискає на кнопку вимкнення сигналізації;

– відправляє на сервер запит на вимикання;

– сервіс керування сигналізацією перевіряє з’єднання з пристроєм та надсилає запит на вимикання сигналізації контролеру пристрою;

– якщо пристрій під’єднаний, то сигналізація успішно вимикається;

– інакше сервіс повідомляє користувача, що пристрій не під’єднано, повертаючи на сторінку керування сигналізацією.

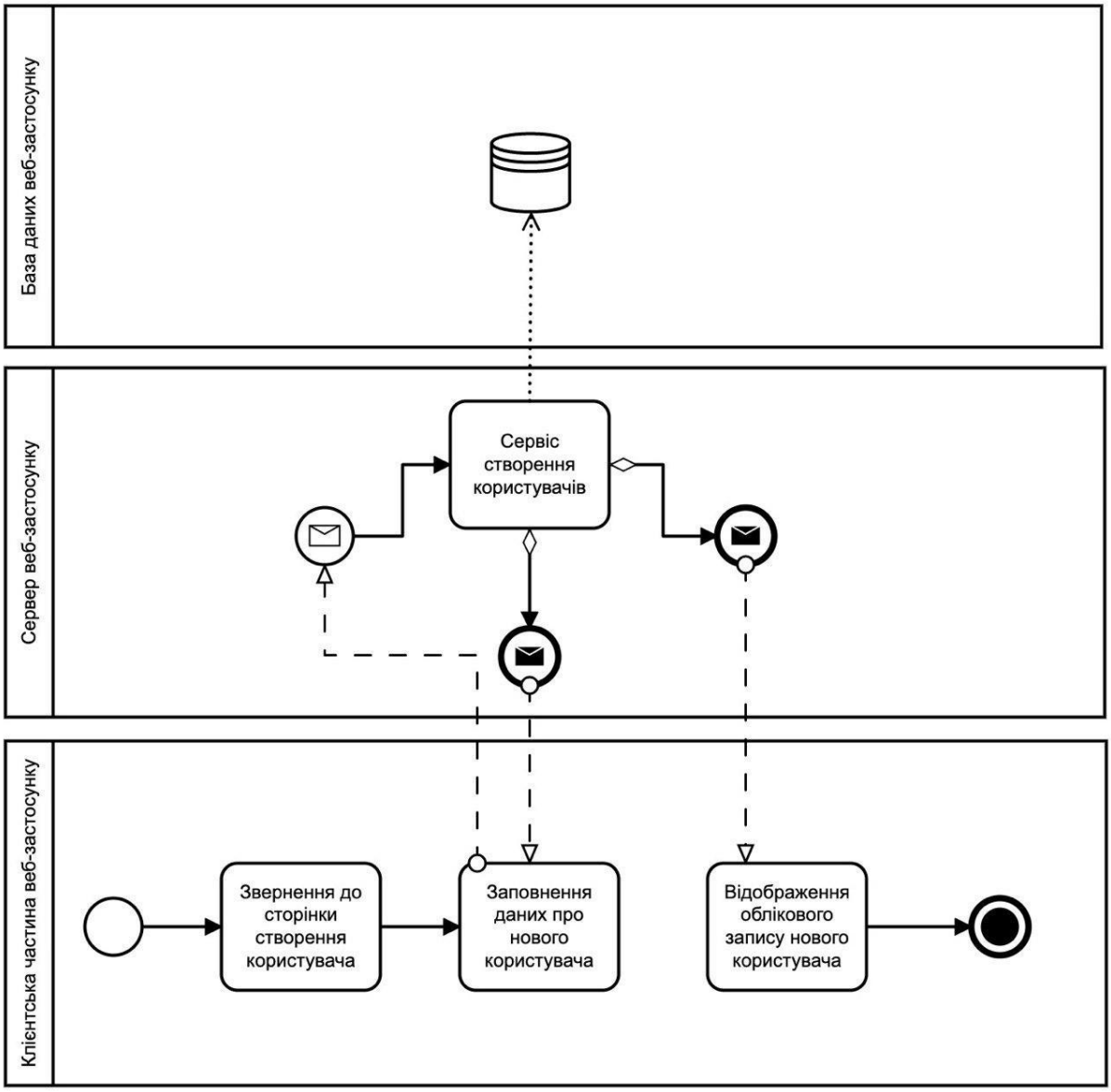


Рисунок 2.5 – Створення користувача

Послідовний опис процесу створення нового користувача:

– користувач звертається до сторінки створення нового користувача;

– користувач заповнює дані про нового користувача;

– дані відправляються на сервер до сервісу створення користувачів;

– якщо дані коректні і користувач, що відправив запит, має роль адміністратора – створюється новий користувач. Його обліковий запис відображається на клієнтській частині;

– інакше повідомляється про помилку створення і сервіс повертає на сторінку створення користувача.

2.2 Архітектура програмного забезпечення

Зобразимо загальну архітектуру системи на Рисунку 2.6

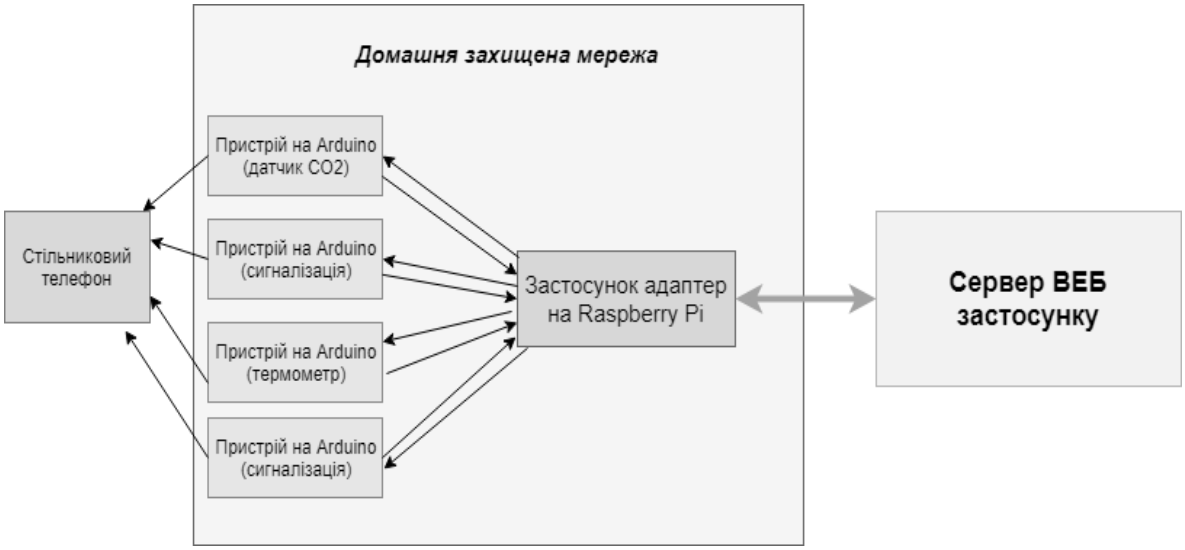


Рисунок 2.6 – Принципова архітектура дипломного проекту

Розроблена система може виконуватись під операційними системами на базі ядра Linux. Сам пристрій працює на базі платформи Arduino.

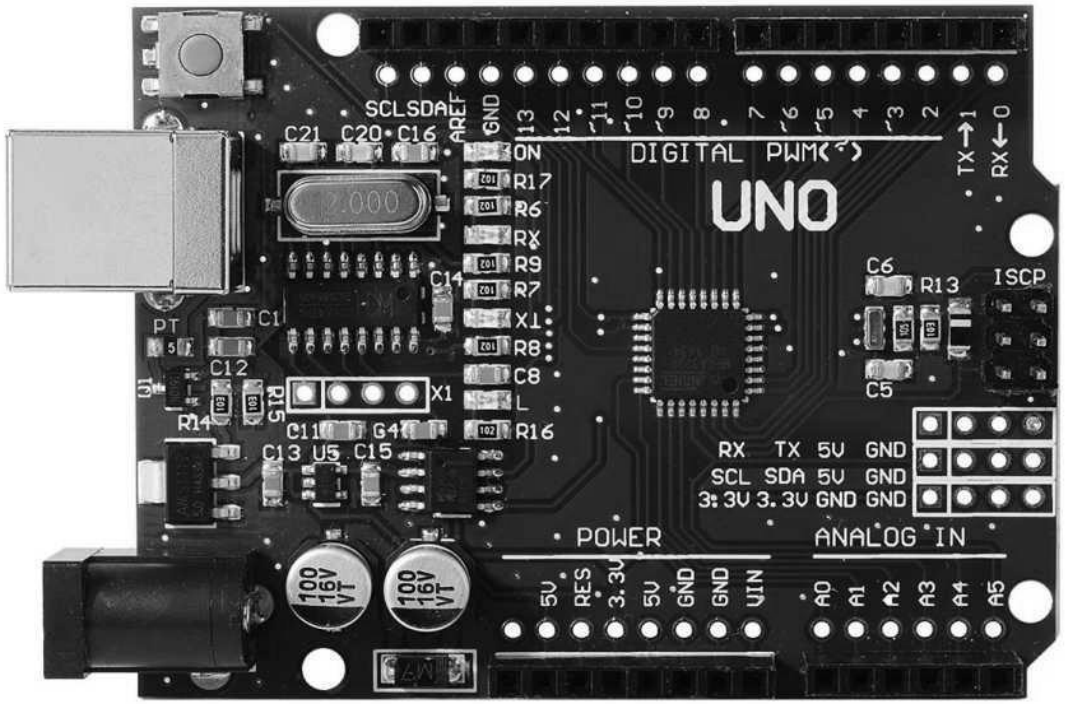


Рисунок 2.7 – Плата Arduino Uno

Усі пристрої під’єднанні до закритої домашньої мережі за допомогою Ethernet shield.

Рисунок 2.8 – Ethernet Shield

Передача SMS повідомлень виконується за допомогою під’єднаного GSM модулю.

Рисунок 2.9 – GSM модуль

Плата Arduino була обрана для зменшення загальної вартості системи, тож необхідно компенсувати малу продуктивність пристроїв адаптером між пристроями та веб-сервером. У ролі адаптера виступає одноплатний комп'ютер Raspberry Pi, який має достатню потужність підтримувати SSL захищене постійне з’єднання з веб-сервером застосунку та обмінюватись http запитами з пристроями у захищеній домашній мережі.

Ідея дешевизни проекту полягає у тому, що для усіх пристроїв у домі ми використовуємо дешеві мікроконтролери Arduino, а незмогу реалізувати ними постійне захищене з’єднання з сервером застосунку компенсуємо лише одним пристроєм адаптером.

Програмний продукт, що задовольняє вимоги системи, повинен бути написаний на платформі, що дає можливості для створення ВЕБ-серверу, підключення до бази даних та створення інтерфейсу користувача.

Мікроконтролер для сигналізації повинен мати змогу розгорнути ВЕБ-сервер для прийому запитів з платформи для керування.

Для організації бази даних була обрана СУБД MySQL, до якої є драйвер від серверної платформи. Система повинна відповідати сучасним стандартам створення ВЕБ-застосунків, тому має бути використана актуальна платформа зі всіма необхідними властивостями, в ролі такої можна взяти Spring.

Spring Framework складається з кількох модулів, які надають широкий спектр послуг:

- контейнер Інверсії управління: Все керування компонентами додатку здійснюється через Інверсію управління;
- аспектно-орієнтоване програмування дозволяє зручно використовувати наскрізний функціонал;
- доступ до даних: робота з реляційною системою управління базами даних;
- управління транзакціями дозволяє об’єднувати кілька API;

– модель-вигляд-управління (model-view-controller): вбудований підхід який розділяє застосунок на три окремі функціональні ролі. Таким чином розробка частин застосунку відбувається незалежно;

– аутентифікація і авторизація: конфігуроване забезпечення системою безпеки, яка підтримує всі основні стандарти, надає проект Spring Security;

– тестування: створений каркас для написання юніт-тестів та інтеграційних тестів.

Для мікроконтролеру пристрою взята апаратна обчислювальна платформа Arduino, основними компонентами якої є плата мікроконтролера з елементами вводу/виводу та середовище розробки Processing/Wiring на мові програмування, що є спрощеною похідною від C/C++.

2.3 Конструювання програмного забезпечення

Опишемо конструктивні складові серверної частини застосування

Таблиця 2.1 – Опис класів (структур) системи

Клас/модуль	**Опис**
HttpHandshakeInterceptor	Клас-перехоплювач для першого http запиту при WebSocket рукостисканні. Використовується для прокидання додаткових даних при рукостисканні
WebSecurityConfigurerAdapter	Клас, що надає базовий функціонал для створення імплементацій налаштувань безпеки
SecurityConfig	Клас-розширення для WebSecurityConfigurerAdapter. Використовується для виняткових налаштувань безпеки системи

Продовження таблиці 2.1

SecurityInitiaizer	Клас, що ініціалізує систему безпеки для додатку
WebSocketConfig	Клас, який містить налаштування для WebSocket з’єднання з додатком-адаптером
AdminController	Клас контролер, що містить обробку запитів для функціоналу адміністраторів
DeviceController	Клас контролер, що містить обробку запитів керування пристроєм сигналізації
Message	Клас, який використовують учасники WebSocket з’єднання для обміну інформацією
NewUserDTO	Клас, екземпляри якого використовуються як об’єкти для обміну даними, а саме для створення нового користувача
AlreadyExistException	Клас, що є нащадком RuntimeException і використовується у сервісі створення користувачів
ResourceNotFoundException	Клас, що є нащадком RuntimeException та використовується для інформування про помилки під час пошуку даних у базі
Role	Клас, що представляє сутність ролі для зберігання у базі даних

Продовження таблиці 2.1

User	Клас, що представляє сутність користувача для його зберігання у базі даних
RoleRepository	Клас, що містить базові методи для роботи с базою даних для сутностей ролей
UserRepository	Клас, що містить базові методи для роботи с базою даних для сутностей користувачів
UserService	Клас, що містить бізнес-логіку для роботи з користувачами
DeviceService	Клас, що містить бізнес-логіку для керування пристроєм сигналізації
Logger	Клас, що містить логіку для роботи з логуванням
CommonUtils	Клас, що містить загальні методи для різних шарів додатку
ServerApplication	Клас, що запускає додаток, викликаючи службові методи фреймворку
ConnectionService	Клас додатку-адаптера, який ініціалізує WebSocket з’єднання з веб-сервером основного додатку
StompSessionHandler	Клас, який займається обробкою подій у рамках STOMP сесії

Розглянемо детальніше кожну функцію по всіх класах в таблиці, що буде наведена нижче

Таблиця 2.2 – Опис методів класів та інтерфейсів системи

Клас/Модуль	**Метод**	**Опис**
HttpHandshakeInterceptor	beforeHandShake(HttpRequest req, HttpResponse resp, WebSocketHandler handler, Map<String, Object> attributes)	Метод, що викликається до моменту, коли рукостискання відбудеться. Приймає запит, відповідь, обробник та атрибути HTTP рукостискання
HttpHandshakeInterceptor	afterHandShake(HttpRequest req, HttpResponse resp, WebSocketHandler handler, Exception ex)	Метод, що викликається після моменту рукостискання. Приймає запит, відповідь, обробник та можливу помилку рукостискання

Продовження таблиці 2.2

WebSecurityConfigurerAdapter	configure(AuthenticationManagerBuilder auth)	Метод конфігурації, який використовує імплементація менеджера аутентифікації за замовчуванням
WebSecurityConfigurerAdapter	getHttp()	Метод, що створює новий об’єкт HttpSecurity або повертає нинішній
WebSecurityConfigurerAdapter	getApplicationContext()	Метод, який повертає контекст веб-застосунку
SecurityConfig	configAuthentication(AuthenticationManagerBuilder auth)	Метод, який налаштовує спосіб аутентифікації та спосіб шифрування паролів для бази даних

Продовження таблиці 2.2

SecurityConfig	configure(HttpSecurity http)	Метод, який налаштовує необхідні ролі для різних контролерів застосунку. Також вмикає додаткові способи авторизації
SecurityConfig	getAuthoritiesQuery()	Метод, що повертає SQL запит до бази даних для вибірки ролей користувача
SecurityConfig	passwordEncoder()	Метод, який повертає кодер для паролів користувачів
SecurityInitiaizer	insertFilters(Filter... filters)	Метод для вставки фільтрів переданих аргументами перед існуючими фільтрами

Продовження таблиці 2.2

SecurityInitiaizer	appendFilters(Filter... filters)	Метод для вставки фільтрів переданих аргументами після існуючих фільтрів
WebSocketConfig	registerStompEndpoints(StompEndpointRegistry registry)	Метод, для реєстрації специфічних кінцевих точок для STOMP
WebSocketConfig	configureWebSocketTransport(WebSocketTransportRegistration registry)	Метод, який виконує конфігурацію налаштувань для обробки повідомлень які відправлені та отримані з WebSocket клієнта
WebSocketConfig	configureClientInboundChannel(ChannelRegistration registration)	Метод для конфігурації каналу зв’язку для вхідних повідомлень

Продовження таблиці 2.2

WebSocketConfig	configureClientOutboundChannel(ChannelRegistration registration)	Метод для конфігурації каналу зв’язку для вихідних повідомлень
WebSocketConfig	configureMessageBroker(MessageBrokerRegistry registry)	Метод, що конфігурує налаштування брокера повідомлень
AdminController	createNewUser(NewUserDTO newUserDTO)	Метод контролера для створення нового користувача
DeviceController	protection(Principal principal, String enableParam)	Метод контролера для увімкнення та вимкнення сигналізації
DeviceController	getAccState(Principal principal)	Метод контролера для запиту поточного стану батареї

Продовження таблиці 2.2

DeviceController	warning(Principal principal)	Метод контролеру, який викликається при спрацюванні сигналізації
DeviceController	subscribe(Map<String, Object> headers, Principal principal)	Метод контролера для підключення застосунка-адаптера до системи
AdminController	deleteUser(String username)	Метод контролера для видалення існуючого користувача
Message	getContent()	Метод, що повертає контент повідомлення
Message	setContent(String content)	Метод, що задає контент для повідомлення

Продовження таблиці 2.2

Message	Message()	Конструктор для класу Message за замовчуванням
Message	Message(String content)	Конструктор для класу Message, що приймає на вхід контент
NewUserDTO	NewUserDTO();	Конструктор для класу NewUserDTO за замовчуванням
NewUserDTO	NewUserDTO(String username, String password);	Конструктор для класу NewUserDTO, що приймає на вхід ім’я користувача та пароль для облікового запису
NewUserDTO	setUsername(String username)	Метод, що задає значення поля username
NewUserDTO	setPassword(String password)	Метод, що задає значення поля password

Продовження таблиці 2.2

NewUserDTO	getUsername()	Метод, що повертає поле username для класу NewUserDTO
NewUserDTO	getPassword()	Метод, що повертає поле password для класу NewUserDTO
AlreadyExistException	AlreadyExistException ()	Конструктор для класу AlreadyExistException за замовчуванням
AlreadyExistException	AlreadyExistException (String msg)	Конструктор для класу AlreadyExistException, що приймає на вхід повідомлення про помилку
ResourceNotFoundException	ResourceNotFoundException ()	Конструктор для класу ResourceNotFoundException за замовчуванням

Продовження таблиці 2.2

ResourceNotFoundException	ResourceNotFoundException (String msg)	Конструктор для класу ResourceNotFoundException, що приймає на вхід повідомлення про помилку
Role	getId()	Метод для отримання значення поля id для об’єкту класу Role
Role	getRole()	Метод для отримання значення поля role для об’єкту класу Role
Role	setId(Long id)	Метод для встановлення значення поля id для об’єкту класу Role
Role	setRole(String role)	Метод для встановлення значення поля role для об’єкту класу Role
Role	Role()	Конструктор для класу Role за замовчуванням

Продовження таблиці 2.2

Role	Role(Long id, String role)	Конструктор для класу Role, що отримує на вхід поля id та role
User	getId()	Метод для отримання значення поля id для об’єкту класу User
User	setId(Long id)	Метод для встановлення значення поля id для об’єкту класу User
User	getUsername()	Метод для отримання значення поля username для об’єкту класу User
User	setUsername(String username)	Метод для встановлення значення поля username для об’єкту класу User
User	getPassword()	Метод для отримання значення поля password для об’єкту класу User

Продовження таблиці 2.2

User	setPassword(String password)	Метод для встановлення значення поля password для об’єкту класу User
User	getSessionId()	Метод для отримання значення поля sessionId для об’єкту класу User
User	setSessionId (String sessionId)	Метод для встановлення значення поля sessionId для об’єкту класу User
User	getRoleSet()	Метод для отримання списку ролей об’єкта класу User
User	setRoleSet()	Метод для встановлення списку ролей об’єкту класу User
RoleRepository	findByRole	Метод для пошуку ролі за її ім’ям
RoleRepository	save(Role role)	Метод для збереження ролі у базі даних

Продовження таблиці 2.2

RoleRepository	saveAll(Iterable<Role> roles)	Метод для збереження списку ролей у базі даних
RoleRepository	findById(Long id)	Метод для пошуку ролі за її ідентифікатором
RoleRepository	existsById(Long id)	Метод повертає булеве значення, яке означає існування ролі з таким ідентифікатором у базі даних
RoleRepository	findAll()	Метод для пошуку всіх ролей в базі даних.
RoleRepository	findAllById(Iterable<Long> ids)	Метод для пошуку всіх ролей в базі даних по заданим ідентифікаторам.
RoleRepository	count()	Метод повертає кількість наявних ролей у базі даних
RoleRepository	deleteById(Long id)	Метод видаляє з бази даних роль з переданим ідентифікатором

Продовження таблиці 2.2

RoleRepository	delete(Role role)	Метод видаляє з бази даних передану роль
RoleRepository	deleteAll(Iterable<Role> roles)	Метод видаляє з бази даних передані ролі
RoleRepository	deleteAll()	Метод видаляє з бази даних всі ролі
UserRepository	findByUsername	Метод для пошуку існуючого у базі даних користувача за його ім’ям
UserRepository	save(User user)	Метод для збереження користувача у базі даних
UserRepository	saveAll(Iterable<User> users)	Метод для збереження списку користувачів у базі даних
UserRepository	findById(Long id)	Метод для пошуку користувача за його ідентифікатором

Продовження таблиці 2.2

UserRepository	existsById(Long id)	Метод повертає булеве значення, яке означає існування користувача з таким ідентифікатором у базі даних
UserRepository	findAll()	Метод для пошуку всіх існуючих користувачів в базі даних.
UserRepository	findAllById(Iterable<Long> ids)	Метод для пошуку всіх існуючих користувачів в базі даних по заданим ідентифікаторам
UserRepository	count()	Метод повертає кількість існуючих користувачів у базі даних
UserRepository	deleteById(Long id)	Метод видаляє з бази даних існуючого користувача з переданим ідентифікатором

Продовження таблиці 2.2

UserRepository	delete(User user)	Метод видаляє з бази даних переданого користувача
UserRepository	deleteAll(Iterable<User> users)	Метод видаляє з бази даних переданих користувачів
UserRepository	deleteAll()	Метод видаляє з бази даних всіх існуючих користувачів
UserService	registerNewUser()	Метод сервісу користувачів для створення нового користувача
UserService	deleteExistingUser()	Метод сервісу користувачів для видалення існуючого користувача
ConnectionService	connect	Метод класу застосунку адаптера для під’єднання пристрою до системи

Продовження таблиці 2.2

DeviceService	protection(String username, String enableParam)	Метод сервісу, який містить бізнес-логіку для увімкнення та вимкнення сигналізації
DeviceService	getAccState(String username)	Метод сервісу, який містить бізнес-логіку для запиту поточного стану батареї
DeviceService	warning(String username)	Метод сервісу, який містить бізнес-логіку дій при спрацюванні сигналізації
StompSessionHandler	afterConnected(StompSession session, StompHeaders connectedHeaders)	Викликається коли сесія готова до використання
StompSessionHandler	handleException(StompSession session, Throwable exception)	Метод, що викликається при помилці сесії
StompSessionHandler	handleTransportError(StompSession session, Throwable exception)	Метод, що викликається при помилці транспортного рівня спілкування

Продовження таблиці 2.2

Logger	log(String msg)	Метод для виведення в лог інформації
Logger	logError(String msg)	Метод для виведення в лог помилки
Logger	logWarning(String msg)	Метод для виведення в лог попередження

Вхідними даними для системи є користувачі та їх запити до API веб-додатку.

Вихідні дані – зібрана статистика дій, котрі відбулися у системі (зміна стану моніторингу, спрацювання сигналізації і т.д.)

2.4 Аналіз безпеки даних

Безпека даних є надзвичайно важливою для цього проекту, тож разом з економією на периферії потрібно потурбуватись про належний рівень безпеки системи.

Перш за все необхідно обрати спосіб хешування паролів в базі. Наразі Argon2 вважається найкращим алгоритмом.

Пристрої на Arduino не мають змоги спілкуватись з веб-сервером через захищене SSL з’єднання, як і підтримувати постійне WebSocket спілкування. Тож, саме для цього необхідний будь-який адаптер, котрий має достатню потужність для підтримки SSL та WebSocket з’єднання. Тепер всі домашні пристрої на Arduino зможуть спілкуватись с адаптером звичайними http запитами всередині закритої домашньої мережі. Для додаткової безпеки, потрібно обмежити доступ до мережі тільки заданими MAC адресами.

2.5 Висновки по розділу

У цьому розділі було розглянуто архітектуру всієї системи, обрано технології для її розробки. Основні процеси використання системи були відображені за допомогою діаграм BPMN. Також були представлені класи та методи програмного забезпечення системи.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

У величезному процесі розробки програмного забезпечення, тестування є тим етапом, на який люди часто не звертають достатньо уваги. Всі припускають, що готова система буде працювати ідеально за вимогами. Але завжди дуже швидко виявляється, що це не так.

Визнаючи важливість та переваги тестування програмного продукту та роблячи його одним з найважливіших етапів при розробці – бізнес сильно збільшує свої шанси отримати у результаті ефективну систему, яка працює без помилок. Отже, для розробки дипломного проекту також необхідний тест план.

У рамках даного плану буде виконано тестування основної частини системи.

У плані будуть перевірені наступні функції:

- створення користувачів;
- авторизація користувачів;
- увімкнення сигналізації;
- вимкнення сигналізації;
- спрацювання сигналізації.

3.2 Підходи до тестування

В рамках плану будуть використані наступні методи тестування:

- компонентне;
- інтеграційне;
- системне.

3.2.1 Компонентне тестування

Методом компонентного тестування будуть перевірені наступні частини застосунку:

- сервісні методи для створення та видалення користувачів;
- методи авторизації.

3.2.2 Інтеграційне тестування

Методом інтеграційного тестування будуть перевірені взаємодії між модулями системи, такі як:

- взаємодія веб-сервера з базою даних;
- взаємодія веб-сервера з застосунком адаптером.

3.2.3 Системне тестування

Методом системного тестування буде перевірено взаємодію найголовніших елементів системи:

- вмикання сигналізації на пристрою;
- вимикання сигналізації на пристрою;
- спрацювання сигналізації на пристрою.

3.3 Критерії проходження тестування

3.3.1 Компонентне тестування

Для компонентного тестування критерієм проходження є успішне виконання кожного пункту тесту. У разі якщо хоча б один пункт не був успішно виконаний – тестування вважається не пройденим.

3.3.2 Інтеграційне тестування

Для інтеграційного тестування критерієм проходження є успішне

виконання кожного пункту тесту. У разі якщо хоча б один пункт не був успішно виконаний – тестування вважається не пройденим.

3.3.3 Системне тестування

Для системного тестування необхідно підключити всі елементи системи та перевірити їх повну взаємодію на коректність поведінки. Тестування вважається пройденим, якщо всі основні сценарії виконуються з очікуваними результатами.

3.4 Процес тестування

3.4.1 Задачі тесту

Тест має перевіряти результат роботи програми у відповідності з очікуваними результатами у тест-кейсах.

3.4.2 План виконання

Спочатку виконується компонентне тестування, для виявлення помилок у відносно невеликих зонах впливу. Потім потрібно провести інтеграційне тестування для перевірки правильної роботи різних частин системи. У разі проходження попередніх етапів необхідно перевірити систему за допомогою системного тестування.

3.5 Вимоги до середовища

3.5.1 Апаратна частина

Апаратна частина має ті самі вимоги, що описані в технічному завданні.

3.5.2 Програмна частина

Для виконання тестування платформа повинна мати операційну систему на базі Linux.

3.5.3 Вимоги до безпеки

Для тестування необхідно мати закриту домашню мережу.

3.5.4 Інструменти

Для виконання тестування використовувати наступні програмні інструменти:

- Postman;
- Google Chrome;
- Стільниковий телефон.

3.6 Опис контрольного прикладу

У якості прикладу можна провести тестування реакції системи на спрацювання сигналізації.

Для підготовки потрібно підключити усі складові до системи, пройти авторизацію на сторінці веб-застосунка та увімкнути сигналізацію.

Тепер необхідно розімкнути магніт на пристрої сигналізації та очікувати SMS повідомлення на стільниковий телефон впродовж десяти секунд.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

4.1.1 Встановлення основного сервісу

Для встановлення веб-застосунку необхідно запустити на сервері побудований за допомогою Apache Maven jar файл.

4.1.2 Встановлення СУБД MySQL

Для встановлення СУБД необхідно перейти на сайт MySQL та завантажити для необхідної платформи. Встановлювати за користувацькою інструкцією.

4.1.3 Здійснення міграції структури БД у СУБД

Для запуску міграції необхідно запустити веб-застосунок на сервері.

4.1.4 Запуск застосунку адаптера

Для запуску адаптеру потрібно запустити побудований jar файл застосунку.

4.1.5 Під’єднання пристрою до живлення

Для роботи пристрою необхідно підключити його до мережі або надати заряджений акумулятор.

4.2 Інструкція користувача

4.2.1 Авторизація у веб-застосунку

Для того, щоб увійти у панель керування пристроєм – користувачу необхідно звернутись до сторінки логіну, заповнити поля входу логін та пароль та натиснути кнопку «Увійти». Якщо користувач правильно вказав дані – він побачить на екрані панель керування.

4.2.2 Увімкнення та вимкнення сигналізації

Для того, щоб увімкнути сигналізацію – користувачу необхідно звернутись до сторінки керування приладом, пройшовши авторизацію. На цій сторінці користувач може натискати на кнопки «увімкнути сигналізацію» та «вимкнути сигналізацію» для вмикання та вимикання.

4.2.3 Створення нового користувача

Для того, щоб створити новий обліковий запис – користувачу необхідно звернутись до сторінки створення облікових записів, пройшовши авторизацію та маючі роль адміністратора. Далі необхідно натиснути на кнопку «Створити обліковий запис» для відкриття форми створення облікового запису. На формі потрібно заповнити необхідні поля з інформацією про нового користувача та натиснути на кнопку «Створити».

4.2.4 Запит поточного стану заряду батареї

Для запиту поточного стану заряду батареї користувачу необхідно звернутись до сторінки керування приладом, пройшовши авторизацію. На сторінці керування приладом необхідно натиснути кнопку «Запит поточного стану батареї».

4.2.5 Вихід з облікового запису

Для виходу з облікового запису користувачу необхідно на будь-якій сторінці натиснути на кнопку «Вихід з облікового запису».

4.2.6 Під’єднання пристрою до системи

Для під'єднання пристрою до системи необхідно закріпити його біля предмету використання, наприклад дверей. Далі потрібно приклеїти магніти на двері та під’єднати пристрій до домашньої мережі за допомогою Ethernet кабелю.

ВИСНОВКИ

При розробці дипломного проекту була проведена всебічна експертиза та аналіз продукту на основі знань різноманітних дисциплін, отриманих під час навчання в університеті. Етапи роботи можна розділити на декілька розділів.

У розділі “Аналіз вимог до програмного забезпечення” був проведений цілісний аналіз предметної області та вимог продукту.

У розділі “Моделювання та конструювання програмного забезпечення” застосунок було змоделювано архітектурно, щоб отримати бачення кінцевого результату та уникнути потенціальних помилок при розробці.

У розділі “Аналіз якості та тестування програмного забезпечення” було розглянуто необхідність тестування програмних продуктів та підходи до тестування дипломного проекту.

У розділі “Впровадження та супровід програмного забезпечення” було описано процес запуску системи у роботу. Також було розроблено інструкцію користувача та адміністратора.

Дипломний проект дав можливість застосувати різнобічні знання задля досягнення кінцевого результату.

ПЕРЕЛІК ПОСИЛАНЬ

1) Spring Framework [Електронний ресурс]. – 2019. – Режим доступу до ресурсу:https://uk.wikipedia.org/wiki/Spring_Framework.

2) Spring Framework documentation [Електронний ресурс]. – 2019. – Режим доступу до ресурсу:https://spring.io/docs.

3) Java 8 documentation [Електронний ресурс]. – 2019. – Режим доступу до ресурсу:https://docs.oracle.com/javase/8/docs/api/.

4) MySQL [Електронний ресурс]. – 2019. – Режим доступу до ресурсу:https://www.mysql.com/.

5) Arduino documentation [Електронний ресурс]. – 2019. – Режим доступу до ресурсу:https://www.arduino.cc/en/Main/Docs.

6) BPMN [Електронний ресурс]. – 2019. – Режим доступу до ресурсу:https://uk.wikipedia.org/wiki/BPMN.

7) EasyEDA [Електронний ресурс]. – 2019. – Режим доступу до ресурсу:https://easyeda.com/ru.

8) GSM [Електронний ресурс]. – 2019. – Режим доступу до ресурсу:https://uk.wikipedia.org/wiki/GSM.

9) Arduino Ethernet [Електронний ресурс]. – 2019. – Режим доступу до ресурсу:https://doc.arduino.ua/ru/hardware/EthernetShield.

10) Arduino Schottky diodes [Електронний ресурс]. – 2019. – Режим доступу до ресурсу:https://www.autodesk.com/products/eagle/blog/schottky-diodes.

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду

GSM сигналізація з веб-сервером

(Найменування програми (документа))

CD-R

(Вид носія даних)

12 арк, 933 Кб

(Обсяг програми (документа) , арк.,) Кб)

config/HttpHandshakeInterceptor

```
package edu.kpi.shulikov.server.config

import org.springframework.http.server.ServerHttpRequest
import org.springframework.http.server.ServerHttpResponse
import org.springframework.http.server.ServletServerHttpRequest
import org.springframework.web.socket.WebSocketHandler
import org.springframework.web.socket.server.HandshakeInterceptor

class HttpHandshakeInterceptor : HandshakeInterceptor {

    companion object {
        const val ATTRIBUTE_SESSION_ID = "sessionId"
    }

    override fun beforeHandshake(request: ServerHttpRequest, response:
ServerHttpResponse, wsHandler: WebSocketHandler, attributes:
MutableMap<String, Any>): Boolean {
        attributes[ATTRIBUTE_SESSION_ID] = (request as
ServletServerHttpRequest).servletRequest.session.id
        return true
    }

    override fun afterHandshake(request: ServerHttpRequest, response:
ServerHttpResponse, wsHandler: WebSocketHandler, exception: Exception?)
{}
}
```

config/SecurityConfig

```
package edu.kpi.shulikov.server.config;

import javax.sql.DataSource;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.Au
thenticationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWe
bSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecur
ityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

  @Autowired
  DataSource dataSource;

  @Autowired
  public void configAuthentication(AuthenticationManagerBuilder auth)
throws Exception {

    auth.jdbcAuthentication().dataSource(dataSource)
        .usersByUsernameQuery(
            "select username, password, 1 as enabled from user where
username = ?")
        .authoritiesByUsernameQuery(getAuthoritiesQuery())
        .passwordEncoder(passwordEncoder());
  }

  @Override
  protected void configure(HttpSecurity http) throws Exception {

    http.authorizeRequests()
        .antMatchers("/admin/**").access("hasRole('ROLE_ADMIN')")
        .anyRequest().authenticated()
        .and()
        .httpBasic().and()
        .formLogin().failureUrl("/error")
```

```
        .usernameParameter("username").passwordParameter("password")
        .and().csrf().disable();
  }

  private String getAuthoritiesQuery() {
    return "SELECT DISTINCT user.username as username, role.role as
authority "
        + "FROM user "
        + "JOIN user_roles ON user.id = user_roles.user_id "
        + "JOIN role ON role.id = user_roles.role_id "
        + "WHERE user.username = ? ";
  }

  @Bean
  public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
  }
}
```

config/SecurityInitializer

```
package edu.kpi.shulikov.server.config;

import
org.springframework.security.web.context.AbstractSecurityWebApplicationIn
itializer;

public class SecurityInitializer extends
AbstractSecurityWebApplicationInitializer {

}
```

config/WebSocketConfig

```
package edu.kpi.shulikov.server.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.messaging.simp.config.MessageBrokerRegistry;
import
org.springframework.web.socket.config.annotation.EnableWebSocketMessageBr
oker;
import
org.springframework.web.socket.config.annotation.StompEndpointRegistry;
import
org.springframework.web.socket.config.annotation.WebSocketMessageBrokerCo
nfigurer;

@Configuration
@EnableWebSocketMessageBroker
```

```
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer
{

  @Override
  public void configureMessageBroker(MessageBrokerRegistry config) {
    config.enableSimpleBroker("/queue", "/topic");
    config.setApplicationDestinationPrefixes("/app", "/user");
    config.setUserDestinationPrefix("/user");
  }

  @Override
  public void registerStompEndpoints(StompEndpointRegistry registry) {
    registry.addEndpoint("/gs-guide-websocket").addInterceptors(new
HttpHandshakeInterceptor());
  }
}
```

controller/AdminController

```
package edu.kpi.shulikov.server.controller;

import edu.kpi.shulikov.server.dto.NewUserDTO;
import edu.kpi.shulikov.server.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class AdminController {

  @Autowired
  UserService userService;

  @RequestMapping(value = "/admin/newUser", method = RequestMethod.POST)
  @Transactional
  public String createNewUser(@RequestBody NewUserDTO newUserDTO) {
    userService.registerNewUser(newUserDTO);
    return "New user was created";
  }
}
```

controller/DeviceController

```
package edu.kpi.shulikov.server.controller;

import edu.kpi.shulikov.server.dto.NewUserDTO;
```

```
import edu.kpi.shulikov.server.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class AdminController {

  @Autowired
  UserService userService;

  @RequestMapping(value = "/admin/newUser", method = RequestMethod.POST)
  @Transactional
  public String createNewUser(@RequestBody NewUserDTO newUserDTO) {
    userService.registerNewUser(newUserDTO);
    return "New user was created";
  }
}
```

dto/Message

```
package edu.kpi.shulikov.server.dto;

import lombok.Data;

@Data
public class Message {

  private final String content;
}
```

dto/NewUserDTO

```
package edu.kpi.shulikov.server.dto;

import javax.validation.constraints.NotEmpty;
import lombok.Data;

@Data
public class NewUserDTO {

  @NotEmpty
  private String userName;

  @NotEmpty
  private String password;}
```

exception/AlreadyExistException

```
package edu.kpi.shulikov.server.exception;

public class AlreadyExistException extends RuntimeException {

}
```

exception/AlreadyExistException

```
package edu.kpi.shulikov.server.exception;

public class ResourceNotFoundException extends RuntimeException {

}
```

model/Role

```
package edu.kpi.shulikov.server.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import lombok.Data;

@Entity
@Data
public class Role {

  @Id
  @GeneratedValue(strategy= GenerationType.AUTO)
  private Long id;

  @Column(unique = true)
  private String role;
}
```

model/User

```
package edu.kpi.shulikov.server.model;

import java.util.Set;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
```

```
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Entity
@Builder
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class User {

  @Id
  @GeneratedValue(strategy = GenerationType.AUTO)
  private Long id;

  @Column(unique = true)
  private String username;

  @Column
  private String password;

  @Column
  private String sessionId;

  @ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.LAZY)
  @JoinTable(name = "user_roles",
      joinColumns = {@JoinColumn(name = "user_id", referencedColumnName =
"id")},
      inverseJoinColumns = {@JoinColumn(name = "role_id",
referencedColumnName = "id")}
  )
  private Set<Role> roleSet;
}
```

repository/RoleRepository

```
package edu.kpi.shulikov.server.repository;

import edu.kpi.shulikov.server.model.Role;
import java.util.Optional;
import org.springframework.data.repository.CrudRepository;
```

```
public interface RoleRepository extends CrudRepository<Role, Long> {

  Optional<Role> findByRole(String role);
}
```

repository/UserRepository

```
package edu.kpi.shulikov.server.repository;

import edu.kpi.shulikov.server.model.User;
import java.util.Optional;
import org.springframework.data.repository.CrudRepository;

public interface UserRepository extends CrudRepository<User, Long> {

  Optional<User> findByUsername(String username);
}
```

service/UserService

```
package edu.kpi.shulikov.server.service;

import edu.kpi.shulikov.server.dto.NewUserDTO;
import edu.kpi.shulikov.server.exception.AlreadyExistException;
import edu.kpi.shulikov.server.exception.ResourceNotFoundException;
import edu.kpi.shulikov.server.model.Role;
import edu.kpi.shulikov.server.model.User;
import edu.kpi.shulikov.server.repository.RoleRepository;
import edu.kpi.shulikov.server.repository.UserRepository;
import java.util.HashSet;
import java.util.Set;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service
public class UserService {

  public static final String DEFAULT_ROLE = "ROLE_USER";

  @Autowired
  private UserRepository userRepo;

  @Autowired
  private RoleRepository roleRepository;

  @Transactional
  public void registerNewUser(NewUserDTO userDto) {
```

```
    if (emailExist(userDto.getUserName())) {
      throw new AlreadyExistException();
    }

    Role role_user = roleRepository.findByRole(DEFAULT_ROLE)
        .orElseThrow(ResourceNotFoundException::new);

    Set<Role> roleSet = new HashSet<>();
    roleSet.add(role_user);

    User newUser = User.builder()
        .username(userDto.getUserName())
        .password(userDto.getPassword())
        .roleSet(roleSet)
        .build();

    userRepo.save(newUser);
  }

  public User findUser(String username) {
    return
userRepo.findByUsername(username).orElseThrow(ResourceNotFoundException::
new);
  }

  private boolean emailExist(String email) {
    return userRepo.findByUsername(email).isPresent();
  }
}
```

adapter/ConnectionService

```
package edu.kpi.shulikov.adapter;

import java.util.Base64;
import java.util.Scanner;
import
org.springframework.messaging.converter.MappingJackson2MessageConverter;
import org.springframework.stereotype.Service;
import org.springframework.web.socket.WebSocketHttpHeaders;
import org.springframework.web.socket.client.WebSocketClient;
import
org.springframework.web.socket.client.standard.StandardWebSocketClient;
import org.springframework.web.socket.messaging.WebSocketStompClient;

@Service
public class ConnectionService {
```

```
  public static final String URL = "ws://localhost:8080/gs-guide-
websocket";

  public static final String USERNAME = "dmytro";

  public static final String PASSWORD = "HEH";

  public static void main(String[] args) {

    WebSocketClient client = new StandardWebSocketClient();
    WebSocketStompClient stompClient = new WebSocketStompClient(client);

    stompClient.setMessageConverter(new
MappingJackson2MessageConverter());

    String plainCredentials= USERNAME + ":" + PASSWORD;
    String base64Credentials =
Base64.getEncoder().encodeToString(plainCredentials.getBytes());

    final WebSocketHttpHeaders headers = new WebSocketHttpHeaders();
    headers.add("Authorization", "Basic " + base64Credentials);

    stompClient.connect(URL, headers, new MyStompSessionHandler());

    new Scanner(System.in).nextLine();
  }


}
```

adapter/ConnectionService

```
package edu.kpi.shulikov.adapter;

import java.lang.reflect.Type;
import java.time.Duration;
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.web.client.RestTemplateBuilder;
import org.springframework.context.annotation.Bean;
import org.springframework.http.ResponseEntity;
import org.springframework.messaging.simp.stomp.StompCommand;
import org.springframework.messaging.simp.stomp.StompHeaders;
import org.springframework.messaging.simp.stomp.StompSession;
import
org.springframework.messaging.simp.stomp.StompSessionHandlerAdapter;
import org.springframework.stereotype.Controller;
import org.springframework.web.client.RestTemplate;
```

```
@Controller
public class MyStompSessionHandler extends StompSessionHandlerAdapter {

  public static final String SECURITY_URL = "http://192.168.0.114:8080/";

  private final Logger logger =
LogManager.getLogger(MyStompSessionHandler.class);

  @Autowired
  private RestTemplate restTemplate;

  @Override
  public void afterConnected(StompSession session, StompHeaders
connectedHeaders) {
    logger.info("New session established : " + session.getSessionId());
    session.subscribe("/user/queue/test", this);
    logger.info("Subscribed to /user/queue/test");
  }

  @Override
  public void handleException(StompSession session, StompCommand command,
StompHeaders headers,
      byte[] payload, Throwable exception) {
    logger.error("Got an exception", exception);
  }

  @Override
  public Type getPayloadType(StompHeaders headers) {
    return Message.class;
  }

  @Override
  public void handleFrame(StompHeaders headers, Object payload) {
    Message msg = (Message) payload;
    logger.info("Received : " + msg.getContent());
    if (msg.getContent().equals("enable=true") ||
msg.getContent().equals("enable=false")) {
      ResponseEntity<String> forEntity = restTemplate
          .getForEntity(SECURITY_URL + "?" + msg.getContent(),
String.class);
      logger.info(forEntity.toString());
    }
    logger.info(msg.getContent() + " was sent");
  }

  @Bean
  public RestTemplate restTemplate(
      RestTemplateBuilder restTemplateBuilder) {

    return restTemplateBuilder
```

```
            .setConnectTimeout(Duration.ofSeconds(3))
            .setReadTimeout(Duration.ofSeconds(3))
            .build();
    }
}
```

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ О.А. Павлов

“___” _____ 2019 р.

GSM сигналізація з веб-сервером

Технічне завдання

КПІ.ІП-5224.045440-02-91

“ПОГОДЖЕНО”

Керівник проекту:

_____К.І. Ліщук

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____Д.Д. Шуліков

Київ – 2019 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: GSM сигналізація з веб-сервером.

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного забезпечення для дистанційного керування домашньою системою сигналізації.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки програмного забезпечення для дистанційного керування домашньою системою сигналізації є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації і управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім.Ігоря Сікорського).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання людьми, які прагнуть захистити доступ до своєї приватної власності.

Метою розробки є зменшення часу реакції на можливі загрози для безпеки приватної власності за рахунок використання мережі Інтернет, а також зменшення собівартості кінцевої системи.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1.1 Для користувача:

– Функціонал для дистанційного керування станом домашньої сигналізації;

– Функціонал для перегляду статистики користуванням.

4.1.2 Розробку виконати на платформі Linux або Mac OS

4.2 Вимоги до надійності

Необхідно забезпечити пристрій акумулятором на випадок вимкнення живлення.

4.2.1 Передбачити перевірку введення інформації.

4.2.2 Передбачити захист від некоректних дій користувача.

4.2.3 Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.2 Обслуговування

Даний продукт потребує регулярного обслуговування у вигляді заміни периферії для пристрою у разі її несправності.

4.3.3 Обслуговуючий персонал

Для обслуговування достатньо одного майстра, котрий зможе констатувати вихід із ладу частин пристрою та зробити їх заміну.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

4.4.2 Мінімальна конфігурація технічних засобів:

Тип процесору x86_64.
Об‘єм ОЗП 2048 Мб.
1 Гб жорсткого диску

4.5 Вимоги до інформаційної та програмної сумісності

4.5.1 Програмне забезпечення повинно працювати під управлінням операційних систем сімейства Unix.

4.5.2 Вхідні дані є набір команд для керування пристроєм сигналізації.

4.5.3 Результатами є SMS повідомлення з інформацією про спрацювання сигналізації.

4.5.4 Програмне забезпечення повинно надавати графічний інтерфейс користувача та REST API для інтегрування в інші сервіси.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред‘являються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

4.8 Спеціальні вимоги

Спеціальні вимоги не пред’являються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Модулі розробленого програмного забезпечення мають бути ретельно задокументовані.

5.2 У склад супроводжувальної документації повинні входити наступні документи:

5.2.1 Пояснювальна записка не менше ніж на 50 аркушах формату А4.

5.2.2 Технічне завдання.

5.2.3 Керівництво користувача (включається в ПЗ).

5.2.4 Програма та методика тестування (включається в ПЗ).

5.3 Графічна частина повинна бути виконана на 3 листах формату А3, котрі включаються у якості додатків до пояснювальної записки:

5.3.1 Схема електрична принципова.

5.3.2 Схема структурна принципова.

5.3.3 Креслення вигляду екранних форм

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

	Назва етапу	Строк	Звітність
1	Вивчення літератури за тематикою проекту	19.03.2019	
2	Розробка технічного завдання	26.03.2019	Технічне завдання
3	Аналіз вимог та уточнення специфікацій	15.04.2019	Специфікації програмного забезпечення
4	Проектування структури програмного забезпечення, проектування компонентів	22.04.2019	Схема структурна програмного забезпечення та специфікація компонентів
5	Програмна реалізація програмного забезпечення	29.04.2019	Тексти програмного забезпечення
6	Тестування програмного забезпечення	06.05.2019	Тести, результати тестування
7	Розробка матеріалів текстової частини проекту	13.04.2019	Пояснювальна записка.
8	Розробка матеріалів графічної частини проекту	20.05.2019	Графічний матеріал проекту
9	Оформлення технічної документації проекту	27.05.2019	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1 Види випробувань

Тестування кінцевого проекту має бути виконано згідно з «Програми та методики тестування».

Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”
В.о. завідувача кафедри
_____ О.А. Павлов
“___” _____ 2019 р.

GSM сигналізація з веб-сервером

Графічний матеріал

КПІ.ІП-5224.045440-03-99

“ПОГОДЖЕНО”
Керівник проекту:
_____ К.І. Ліщук

Нормоконтроль:
_____ К.І. Ліщук

Виконавець:
_____Д.Д. Шуліков

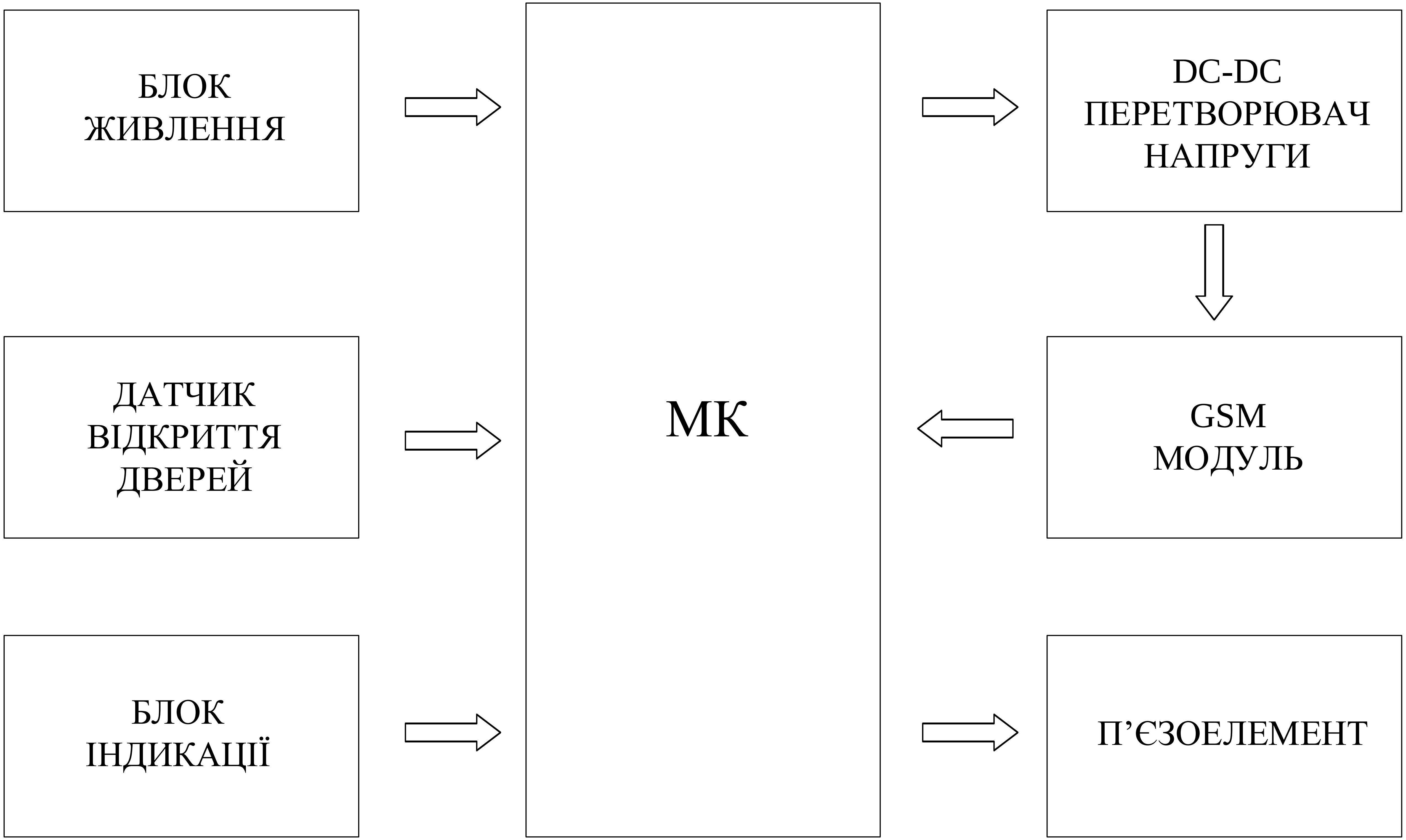
Київ – 2019 року

КПІ.ІП-5224.045440.03.99 СП

U2
18650x2 Batter Box
B-
B+
7.4
U3
HX-2S-JH10 BMS board
P-
P+
BM
P1
WJ2EDGVC-5.08-2P
VCC
GND
BUZZER1
BUZZERHMB1206-03
R9
10k
R1
220
OUT
U6
LM2596-MODULE
IN+
IN-
OUT+
OUT-
LM2596MODULE
U5
SIM800L
GSM
RING
DTR
MICP
MICN
SPKP
SPKN
VCC
RST
RX
TX
GND
C3
2200uF
U4
arduino_nano rs485
D13
3V3
REF
A0
A1
A2
A3
A4
A5
A6
A7
5V
RST
GND
Vin
D12
D11
D10
D9
D8
D7
D6
D5
D4
D3
D2
GND
RST
RXD
TXD
R2
10k
R3
10k
R4
10k
R7
1k
Q1
BD137
DC1
DC-005_1
C2
2200uF
U1
LED4
φ5 RGB Transparent CK
R8
220k
R5
10k
R6
10k
D1
VS-12CWQ03FN-M3

					КПІ.ІП-5224.045440.03.99 СП	
Зм.	Арк.	№ документа	Підпис	Дата	Схема електрична принципова	Літера / Маса / Масштаб
Розробив		Шуліков Д.Д.				
Перевірив		Ліщук К.І.				Аркуш / Аркушів
Т. кон.						
Н. кон.		Ліщук К.І.			GSM сигналізація з веб-сервером	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-52
Затвердив		Ліщук К.І.				

КПІ.ІП-5224.045440.03.99 СС



					КПІ.ІП-5224.045440.03.99 СС			
Зм.	Арк.	№ документа	Підпис	Дата		Літера	Маса	Масштаб
Розробив		Шуліков Д.Д.			*Схема електрична структурна*			
Перевірив		Ліщук К.І.						
Т. кон.						Аркуш		Аркушів
					GSM сигналізація з веб-сервером	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-52		
Н. кон.		Ліщук К.І.						
Затвердив		Ліщук К.І.						

Monitoring

TURN ON TURN OFF

Battery state: 100%

UPDATE

WEB-BASED GSM ALARM

log out

Monitoring

TURN ON TURN OFF

Battery state: 100%

UPDATE

Statistics

Date	Time	Action
01.02.2019	22:30	Turn off monitoring
01.02.2019	12:35	Disturbance
01.02.2019	12:30	Disturbance
01.02.2019	07:30	Turn on monitoring
30.01.2019	20:04	Turn off monitoring
30.01.2019	06:40	Turn on monitoring
29.01.2019	19:12	Turn off monitoring
29.01.2019	07:42	Turn on monitoring
28.01.2019	20:17	Turn off monitoring
28.01.2019	07:30	Turn on monitoring

Date	Time	Action
01.02.2019	22:30	Turn off monitoring
01.02.2019	12:35	Disturbance
01.02.2019	12:30	Disturbance
01.02.2019	07:30	Turn on monitoring
30.01.2019	20:04	Turn off monitoring
30.01.2019	06:40	Turn on monitoring
29.01.2019	19:12	Turn off monitoring
29.01.2019	07:42	Turn on monitoring
28.01.2019	20:17	Turn off monitoring
28.01.2019	07:30	Turn on monitoring

Please sign in

Username

Password

Sign in

log out

					КПІ.ІП-5224.045440.03.99 КЕ	
Зм.	Арк.	№ документа	Підпис	Дата	Креслення вигляду екранних форм	Літера / Маса / Масштаб
Розробив		Шуліков Д.Д.				
Перевірив		Ліщук К.І.				Аркуш / Аркушів
Т. кон.						
Н. кон.		Ліщук К.І.			GSM сигналізація з веб-сервером	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-52
Затвердив		Ліщук К.І.				